ISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

Assessment and Development of Software Engineering Tools

January 1991

ASQB-GB-90-015

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30,

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
NONE

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE
N/A

3. DISTRIBUTION / AVAILABILITY OF REPORT
N/A

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
ASQBG-I-90-015

5. MONITORING ORGANIZATION REPORT NUMBER(S)
N/A

6a. NAME OF PERFORMING ORGANIZATION
Georgia State University

6b. OFFICE SYMBOL (if applicable)

7a. NAME OF MONITORING ORGANIZATION
N/A

6c. ADDRESS (City, State, and ZIP Code)
University Plaza
ATTN: Dep of Math & Comp Science
Atlanta, GA 30303

7b. ADDRESS (City, State, and Zip Code)
N/A

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
AIRMICS

8b. OFFICE SYMBOL (if applicable)
ASQB - GI

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)
115 O'Keefe Bldg.,
Georgia Institute of Technology
Atlanta, GA 30332-0800

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62783A	DY10	02-04-01	

11. TITLE (Include Security Classification)
Assessment and Development of Software Engineering Tools (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Sue Conger; Martin Fraser; Ross Gagliano; Kuldeep Kumar; Ephraim McLean; G. Scott Owen

13a. TYPE OF REPORT

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT (Year, Month, Day)
1991, January, 16

15. PAGE COUNT
31

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
CASE; Intelligent TestBed; Vienna Development Method; 2167A; 2168; RSC_MGT System; Software Development Environment; Process Description Language; Boehm; Davis/Olson; Waterfall Model; Functional Decomposition; OOD

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

The research project began with the feasibility study and preliminary design of the Intelligent TestBed (ITB). The final technical report describes the motivation for, as well as the design and development of, the ITB. The goal of the ITB is to provide a capability to categorize and associate phases of the systems development life cycle with software methodologies and Computer-Aided Software Engineering Tools. By automatically linking phases to tools, the ITB could benefit two different groups of users. First, researchers and developers can determine the availability, feasibility and similarity of software tools that track systems development and maintenance. Second, managers and analysts, who may have no particular preferences for methods or tools, will be able to assimilate advancements in the several fields in addition to possessing the means to compare available products.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Howard C "Butch" Higley

22b. TELEPHONE (Include Area Code)
(404) 894-3110

22c OFFICE SYMBOL
ASQB-GI

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Computer-Aided Software Engineering (CASE) tools and their proper use is a major concern of the Information Systems Engineering Command. As such, a technical study was performed by Georgia State University, under close direction of AIRMICS, to devise a method by which the software developer could be assured of choosing the correct tool for his particular application. A prototype was derived from the method and was built to show the functionality of the method. The prototype tool is called the Intelligent Testbed (ITB) and is available for demonstration by AIRMICS.

The enclosed final report provides an understanding of the method and the ITB. AIRMICS has attempted to direct the focus of the report on the features to be gained by the use of the tool and method and not so much on the technical details. As such, the appendices have been withdrawn but are available upon request. Your comments on all aspects of the document are solicited.

This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited, and is not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ Glenn E. Racine
Chief
CISD

s/ John R. Mitchell
Director
AIRMICS

Final Research Report

ASSESSMENT AND DEVELOPMENT OF SOFTWARE ENGINEERING TOOLS

Martin D. Fraser, Ross A. Gagliano, G. Scott Owen,

Department of Mathematics and Computer Science

and

Sue A. Conger, Kuldeep Kumar and Ephraim R. McLean

Department of Computer Information Systems

GEORGIA STATE UNIVERSITY

Atlanta, GA 30303

Prepared for

U. S. Army Institute for Research in Management
Information, Communications and Computer Sciences (AIRMICS)
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332

Contract Number: DAKF11-89-C-0014
Contract Period: 2/16/89 - 3/30/90

Project Overview

This Research Report describes the work accomplished under Contract DAKF11-89-C-0014 for the U. S. Army Institute for Research in Management Information, Communications and Computer Sciences (AIRMICS) by Georgia State University (GSU). The research was carried out by the following six principal investigators at GSU: Drs. Martin D. Fraser, Ross A. Gagliano, and G. Scott Owen of the Department of Mathematics and Computer Science within the College of Arts and Science, and Drs. Sue A. Conger, Kuldeep Kumar, and Ephraim R. McLean of the Department of Computer Information Systems within the College of Business Administration. Dr. Gagliano served as Project Director with Dr. Owen as Associate Project Director.

This is the final technical report on this project, consisting of three main sections and four appendices. The appendices contain copies of two papers and two user guides. The papers were published in the Proceedings of, and presented at, the Eighth Ada Technology Conference held in Atlanta during March 5 - 8, 1990. In the main body of the report are contained discussions of the salient issues involved in this research with appropriate related topics suggested for future investigation included in a future activities section.

The views and conclusions in this report are those solely of the investigators and should not be interpreted as representing official policies, expressed or implied, of AIRMICS, the Department of the Army, Georgia State University, or other government agencies.

On behalf of all of the GSU investigators, an expression of sincere appreciation is extended to both the AIRMICS Division Chief, Mr. Glenn Racine, and to the Technical Monitor, Mr. Howard (Butch) Higley, for their constant support during this project.

Table of Contents

Administrative

Under contract DAKF11-89-C-0014 which became effective on 16 February 1989, six faculty investigators and two graduate students at Georgia State University (GSU) participated in the research project described in this report. The focus of the research is software engineering tools, and the technical sponsor of the project is the U. S. Army Institute for Research in Management Information, Communications and Computer Sciences (AIRMICS) located on the Georgia Tech campus in Atlanta. Contractual arrangements were provided through the U. S. Army Forces Command (FORSCOM) at Fort McPherson, GA.

Each investigator contributed to all phases of the project, although each received support (one teaching course release) during only one academic quarter. The actual amount of time that each investigator spent on the project varied over the contract period, but the total time spent on the project per investigator exceeded the release time (nominally, one person-month or about 170 hours of effort pro-rated over approximately 13 months). Two investigators (Fraser and Gagliano) received support during one quarter of the calendar year 1989, while the other four received support only during the summer quarter 1989.

The project was originally scheduled for a ten-month period; however, a no-cost contract extension was requested and granted such that the project was extended to thirteen months (until March 30, 1990). Thus, a total of thirteen monthly Progress Letters were submitted in accordance with the amended contract.

In accordance with project and contract requirements, several In-Progress-Reviews (IPRs), both formal and informal, were conducted. A chronology of the significant project events and activities is provided in the next (Project Accomplishments) section.

Project Accomplishments

The major research accomplishments during this contract period were as follows:

Project Organization and Management

1. Formulated an overall project plan for task accomplishment and specific investigator assignments;
2. Conducted periodic organizational meetings;
3. Conducted weekly technical meetings;
4. Held preliminary meeting at AIRMICS in November 1988;
5. Conducted first IPR on June 9, 1989 at AIRMICS;
6. Presented overview of the project at the AIRMICS/MMES Reusability/Metrics Workshop on June 20, 1989 at the Pierremont Plaza Hotel in Atlanta;
7. Held second IPR on October 27, 1989 at AIRMICS;
8. Hosted AIRMICS personnel at GSU and conducted a demonstration of the ITB on the MAC IIcx and on the IBM PS/2 in November 1989;
9. Provided concepts and ideas to AIRMICS personnel in early 1990 for presentation to the new DISC4 (LTG Hilmes) at an anticipated March 1990 Atlanta visit; and
10. Briefed Dr. John Michael Palms, new GSU President, on AIRMICS and results of this project, as well as other prior but related computing research during January 1990.

Literature Survey

1. Initiated literature surveys and requests for materials on the various topics and technical issues involved as indicated below;
2. Reviewed articles specifically on CASE tools, software reuse, formal software specification and design methods, the Vienna Development Method (VDM), and Function Points Analysis (FPA);
3. Compiled list of references (attached); and
4. Obtained copies of DoD Software Standards 2167A and 2168.

Conferences, Meetings and Symposia

1. Attended the Eastern Multi-Conference sponsored by the Society for Computer Simulation (SCS), the IEEE and the ACM in Tampa, FL in March 1989;

2. Attended the 11th International Conference on Software Engineering in Pittsburgh, PA in May 1989;

3. Visited the Software Engineering Institute (SEI) at the in Pittsburgh, PA in May 1989;

4. Visited several agencies in the Washington, DC area to describe the research project and the ITB in November 1989; and

5. Attended 8th Ada Tech Conference, March 1990, Atlanta.

Equipment and Software

1. Obtained copies of, and demonstrated, commercial CASE products (e.g., KnowledgeWare, Excelerator, IEW, and BriefCASE);

2. Obtained State approval for the procurement of, and ordered and subsequently received, the Alsys Ada compiler for the GSU Sun workstations;

3. Ordered, received, and tested an RGB-to-video converter system (board and adapter) to allow the recording of demonstrations of systems such as the ITB and RSC_MGT;

4. Obtained State approval for the procurement of, and ordered, received and installed the MAC IIcx; and

5. Installed several demonstration packages on the MAC IIcx.

Software Tools

1. Prepared re-design plan for GSU RSC_MGT System as a software tool;

2. Performed the preliminary analysis of the design of the ITB, a prototype of which was built as a software tool to provide tool needs-assessment;

3. Planned, scripted, and subsequently produced a master videotape copy of a session demonstrating the ITB;

4. Made distribution copies of the videotape of the demonstration session of the ITB; and

5. Distributed copies of the ITB videotape to AIRMICS, the GSU administration, and selected federal agencies in the Washington, DC area.

Papers

1. Prepared an invited working paper on the design of the GSU RSC_MGT System and its multidimensional scaling implementation using conceptual closeness;

2. Prepared and submitted a joint paper for the 9th IEEE Phoenix Conference on the ITB;

3. Prepared and submitted two papers for the 8th Ada Technology Conference (Atlanta during March 1990);

4. Prepared and sent abstracts on the ITB to the 1990 ACM 18th Annual Computer Science Conference (in Washington, DC during February 1990);

5. Received acceptance of two papers for the 8th Ada Tech Conference, and completed the technical work on the two presentations; and

6. Gave two presentations (the first on the use of the ITB for Software Engineering instruction and the second on VDM and the relationships between formal methods and structured techniques; copies of both papers are included in the appendices) for the 8th Ada Tech Conference which was held at the Hyatt Regency in Atlanta, March 5-8 1990.

Assistance to Other Agencies

1. Asked by Mr. Robert Holibaugh of the SEI to provide some assistance on the use of VDM;

2. Contacted by Mr. Ted Ruegsegger of SofTech, Inc., which is under contract to the RAPID Center, to provide copies of the GSU EFISS paper (October 1988) and other information on the GSU RSC_MGT System;

3. Contacted by Mr. John Atkins of West Virginia University (also with the NASA-funded AdaNet project) to provide information on the GSU RSC_MGT System;

4. Sent informatio, to Mr. Atkins on the GSU RSC_MGT System, who subsequently called to set up a visit at Morgantown, WV to discuss the GSU implementation of the Faceted Classification Scheme (FCS) and the conceptual closeness measure (trip planned for October 1989);

5. Completed coordination of the visit to Morgantown to provide briefing on GSU implementation of the FCS with the conceptual closeness measures in the GSU RSC_MGT System (trip was later postponed pending expenses being paid by NASA);

6. Sent materials (EFISS paper of October 1988 and information on the GSU RSC_MGT System) to Mr. Ruegsegger of SofTech, Inc./RAPID Center in Waltham, MA; and

7. Sent requested copies of papers on the ITB to the University of South Alabama, Auburn University, the University of Alabama, and Clemson University.

Products

GSDM

1. Developed and refined the Georgia State Development Methodology (GSDM) to derive the Georgia State Development Life Cycle (GSDLC) from four standard system application development methodologies: IEEE, DoD Standard 2167A, Boehm, and Davis/Olson; and

2. Completed GSDM and GSDLC.

ITB

1. Initiated and completed the design of the prototype ITB;

2. Implemented the prototype ITB on an IBM PS/2 Model 80 using Guide 2.0 hypertext under Microsoft Windows/386;

3. Began populating the prototype ITB with software methodologies and references of example CASE tools;

4. Undertook second phase of the project which included porting the ITB to both a PS/2 Model 70 and a Mac IIcx (ordered specifically for the project);

5. Completed the installations of the several prototype systems of the ITB, which was done on an IBM PS/2 Model 80 (using Guide 2.0 Hypertext under MS Windows/386), and successfully ported to the Mac IIcx (also using Guide); and

6. Wrote user documentation for the ITB for both the IBM PS/2 systems, and the Macintosh IIcx.

RSC_MGT System Re-tool

1. Documented the designed of the multi-dimensional scaling (MDS) implementation of the conceptual closeness technique in the GSU RSC_MGT System; and

2. Began conversion of the GSU RSC_MGT system for operation on a mainframe, the GSU System D (Amdahl).

Formal Specifications

1. Reviewed current status of PSL/PSA and analyzed its relationships to VDM; and

2. Developed stepwise refinement process of SADT using VDM.

Project Background

The research project began with a feasibility study and preliminary design of the Intelligent Test Bed (ITB). This report describes the motivation for, as well as the design and development of, the ITB. The goal of the ITB is to provide a capability to categorize and associate phases of the systems development life cycle with software methodologies and Computer Aided Software Engineering (CASE) tools. By automatically linking phases to tools, the ITB could benefit two different groups of users. First, researchers and developers can determine the availability, feasibility and similarity of software tools that track systems development and maintenance. Second, managers and analysts, who may have no particular preferences for methodologies or tools, will be able to assimilate advancements in the several fields in addition to possessing the means to compare available products.

The research project consisted of four related tasks: (1) the analysis of the needs for, and the development of, software tools for demonstration on a prototype software maintenance workstation; (2) a review of potential Reusable Software Component (RSC) and Computer-Aided Software Engineering (CASE) tools; (3) a preliminary investigation into the appropriateness of the Vienna Development Method (VDM) for software requirements and specification analysis; and (4) an implementation of a prototype ITB which is an example of a software tool that has application for both software tool needs assessment and analysis.

Introduction

Software Methodologies

A software development methodology is generally considered to be a comprehensive set of procedures, tools and techniques. Such a set of items within a methodology is designed to work together to assist in the completion of a software system. In the literature of an estimated 500 articles on software methodologies, there are many articles in which comparisons between various methodologies are given. However, no evidence has been found that anyone has attempted to categorize them exhaustively, either by author (associated with a methodology) or by phase of the development life cycle. Such a categorization would obviously be very tedious, expensive, and time-consuming, not to mention the fact that it would run the risk of being obsolete as soon as it was completed.

A software specification or design representation form is a schema for modeling, or describing, a system through its various stages of development [FREE80]. During systems development, developers strive to move a system from its current (real or object) state towards a representation (or model) of the final state of the system. This is usually accomplished through diagnosis and manipulation of the model of the desired system which is then submitted for implementation.

Subsequent system implementation is, therefore, a set of sequential transformations from an abstract description of the desired (or designed) system through successively more concrete manifestations to a final system which becomes the installed version of the system. Representation forms, correspondingly, provide a schema or template which is used by the designers to define both the syntax and the semantics of the system model.

Software and CASE Tools

In the context of software development methodologies, a software tool is any physical device or concept that helps a developer or analyst perform development tasks. These tasks may include analysis (understanding the current system, verification of the understanding, or diagnosis), specification, design (logical, external or physical), implementation and operation of the system. Tools can be either manual (principally by pencil and paper) or automated (tools, dictionaries, repositories, or environments). Specialized environments are known as Software Development Environments (SDEs). Some particular software tools are called CASE tools.

A CASE, or Computer Aided Software Engineering, tool is an automated tool, facility or SDE that is used in the production, enhancement, or maintenance of software [SUYD87]. Through CASE tools, many of the manual representation forms have been automated and whose development is guided by an appropriate applications methodology. The categorization of such tools would provide valuable insight to potential productivity gains from using specific CASE tools.

Of course, this may be premature at the present time because there are still many difficulties with CASE tools. The initial CASE tool market was strictly for new applications development; thus, few CASE tools, if any, are specifically for software migration, enhancements, or maintenance. Moreover, despite the hundreds of CASE tools that have been advertised, it has been reported that less than 10% of the mainframe users have actually purchased such tools. Additionally, of those that purchased CASE tools, only 1 to 2 % actually have them in use, and where 80% describe their use as experimental only [CONG89].

The original intent of this project was to concentrate on CASE tools that appear to be the most promising for applicability to the Standard Army Management Information Systems (STAMIS). Early in the project, however, it was recognized that the most significant class of such tools are commerical CASE tools with only the potential for specific STAMIS applications. Also, inasmuch as other GSU research has involved software reuse and functional decomposition [GAGL88a, GAGL88b, GAGL89, OWEN87, OWEN88a, OWEN88b], which are two important related applications, several additional prospective tools are described in this report.

Other Approaches

There are other approaches for the comparison of methodologies and tools. For example, there are Process Description Languages (PDLs), Software Tool Kits (STKs), and the SDEs that were previously mentioned.

Process Description Languages (PDLs). PDLs can be used to describe the relationship between software development and associated tools. PDLs can correlate both intermediate and end products by phase of the life cycle. However, a PDL is aimed less at organizing and analyzing existing methodologies than to provide: 1) formal, usually algebraic, mechanisms for describing a methodology; and 2) means for generating (adapting from existing scripts) specific methodologies to suit a particular development context.

By contrast, the ITB is designed to organize and analyze tools by existing methodologies and particular phases of the life cycle supported. The PDL approach, on the other hand, is used for describing and developing customized methodologies for specific environments. Though both are based on life cycle phases, they are supported from different bases (formal description for PDL versus informal for ITB), and intended for different ends (SDE generation for PDL versus phase/methodology/tool relationships for the ITB).

Software Tool Kits (STKs). Another related approach involves the development of a Software Tool Kit (STK), the first of which that was proposed was called the "Analyst-Tool-Kit." An STK is a collection of methodologies, tools and techniques from which a developer could select appropriate components as needed. Unlike the PDL approach or the ITB, the STK does not have an overall organizing scheme. Like the ITB, however, it is based upon an informal descriptions of the methodologies and tools, but as yet no automated support for STKs have been proposed.

Software Development Environments (SDEs). Two SDEs are IS/DSS and Methodology Engineering$_R$. The concept of Information System / Development Support System (IS/DSS) was first proposed [WELK83] as a method for defining and generating customized systems development methodologies and automated systems development support. Again, like PDL, it provides a formal approach to describe methodologies based largely on General Systems Theory. Some IS/DSS concepts have been used in defining methodologies for both office automation, and the European Economic Community ESPRIT project [SCHA88].

Methodology Engineering is an extension of IS/DSS [KUMA88] which, again like PDL, attempts to generate customized methodologies and automated support for specific systems development contexts. The development of Methodology Engineering is currently underway at both GSU and a commercial firm, MetaSystems, Inc.

GSU Development Methodology

Prior to developing the ITB, it was necessary to identify its dimensions and select certain applicable phases of the software life cycle. That approach was called the Georgia State Development Methodology (GSDM), and it is an attempt to provide answers to key questions; e.g., which life cycle is the most appropriate, how are phases defined, and how can tools and methodologies be related?

Four of the major system development life cycles were considered: the DoD Standard 2167A [D2167A], the Boehm model [BOEH81], the IEEE Standard Life Cycle [IEEE83], and the Davis/Olson Life Cycle [DAVI85]. A reference life cycle was proposed, called the Georgia State Development Life Cycle (GSDLC) that would correlate various components of the four.

Accordingly, under the rubric of GSDM, generic steps in both the system and software development life cycles were specified in the GSDLC. These steps (see Table I) are specified as follows: Initiation, Problem Definition, Feasibility, Analysis, Conceptual Design, Detailed Design, Coding, Testing, Installation, Operation, and Maintenance. The definitions and descriptions of each of these steps are detailed in the Appendices.

Table I. Life Cycle Comparisons

GSDM	DoD 2167A	Boehm	IEEE	Davis/Olson
Sys Init			SLC-Req	
Proj Defn			SLC-REQ	Defn Proj
Feasibility		Feasibility	SLC-Req	Defn-Feas
GSDLC Anal	SRA	Req	DLC-Req	Defn-RA
" Conc Des	Prel Des	Prod Des	DLC-Des	Defn-C Des
" Design	Det Des	Det Des	DLC-Des	Dev-Phy Sys
" Code/Unit Test	Code/CSU Test	Code	DLC-Impl	Dev-Prg Dev
" Testing	CSC/Integ/ CSCI Test	Integ	DLC-Test	Dev-Prod Dev
" Instl Ckout	Sys Integ	Impl	DLC-Impl	Instl-Conv/ Testing
Oprn/Maint		Maint	SLC-Oprn Maint	Instl-Post Audit
Retirement		Phaseout	SLC-Retire	Instl-Post Audit
Continuing Activities -- All Phases				
V & V	Review	V & V		Mngt Rev
CM	Not Clear	CM		

Legend

Conc = Conceptual
CM = Configuration Management
CSU = Computer Software Unit (2167A)
CSC = Computer Software Component (2167A)
CSCI = Computer Software Component Integration (2167A)
Defn = Definition
Des = Design
Det Detailed
DLC = Development Life Cycle (IEEE)
Integ = Integration
Mngt = Management
SLC = System Life Cycle (IEEE)
Prel = Preliminary
Proc = Procedure
Prod = Product
Rev = Review
V & V = Verification and Validation

The Intelligent TestBed (ITB)

Background

The motivation for much of the research in this project really began as an extension of a previous software reusability project that, although conducted for AIRMICS, was managed through the Martin Marietta Energy Systems (MMES) Company. The purpose of the earlier MMES project was to develop and/or investigate prototype software tools specifically for the creation, storage, and retrieval of Ada Reusable Software Components (or Ada RSCs). Such tools then would assist in the shift of RSC applications "to the left" (or earlier) in the software development life cycle. In other words, there was a strong indication that features of reusability should encompass more of the software development life cycle such as requirements, specifications, and design, and not simply the implementation or coding phases.

It was envisioned that these tools would normally interface with other software tools or RSCs through software library systems. The development of the GSU RSC_MGT System took on greater significance, therefore, during its prototype implementation. Moreover, in that same MMES project, approximately fifty commercial software firms engaged in embedded systems development were canvassed to determine the extent to which they employed RSC libraries, Ada development facilities or CASE tools. The results from this survey indicated that a far greater need existed than anticipated for the <u>means</u> to both **describe** actual or proposed tools (RSC, CASE or other) and to **assess** the ability of the tools to solve MIS problems, especially in an Ada environment.

It was from this need that the idea for the ITB emanated. The basic function of the ITB would be to categorize software tools as well as associate software development life cycle phases with design methodologies. The long range goal of the ITB was for it to facilitate the analysis of candidate software tools and provide a means to test "proofs of concept" (the <u>testbed</u>) for actual tool specifications. Other future roles of the ITB would allow for: its use in the development of tutorials on software tools; and the introduction of an expert system to guide development (the <u>intelligence</u>). Fully functional, the ITB would provide answers on how to: evaluate a set of candidate tools; find the right tools for particular instances; and determine the compatibility between various tools.

ITB Development

The original approach to the design of the ITB was to simply build an automated matrix generator which would relate the data base of existing tool characteristics or specifications of potential but non-existent tools (rows of the matrix) to the phases of the software life cycle (columns). This information could then be presented in both graphical and textual formats; i.e., through the use of Hypertext.

Such a matrix would indicate the relative value of candidate tools in the life cycle based on tool characteristics. Knowledge of currently used tools would indicate what is being used and how it is used. Next, current tools, methods and techniques in use, say, by the Army would be mapped to this matrix. Similarly, existing tools not currently in use would be surveyed and mapped to a second matrix. The ITB could then provide functions for automatically comparing these matrices to identify specifically lacking areas.

Nonetheless, it was felt that the long range goal of the ITB was that it should not be just a database of software tools. Thus, almost from the very beginning, it was decided that the focus of this part of the research should be the prototyping of a system with the potential for an intelligent front-end. Moreover, this prototype should provide an indication of the potential for the development of a system which accepts desired attributes from the life cycle and matches them with features of either available or potential tools.

Three different audiences could gain useful information into methodologies and CASE tools through the ITB. First, researchers and developers could determine the availability, feasibility and similarity of software tools that track systems development and maintenance. Second, managers and analysts would have a basis to compare available CASE products, and a means to comprehend advancements in the several fields. Third, instructors could use the ITB to promote greater understanding of the relationships between different methodologies, CASE tools and development phases.

After considerable discussion, it was decided that the phases of the GSDLC should be one of the major dimensions of the ITB. Three methodology classes was chosen as the other dimension. Recall that four main life cycles (Boehm's Waterfall, IEEE Standard, Davis/Olson MIS approach, and the DoD Std 2167A) were combined to form the more generic GSDLC.

ITB Specification and Implementation

The design of the ITB can be efficiently explicated in a formal specification language [DUCE88]. The Vienna Development Method (VDM) was chosen, and a top-level domain specification of the ITB was then prepared in the VDM language META-IV [JONE78, BJOR78]. A VDM specification of the ITB is contained in appendix E.

The ITB development used a Guide 2.0 hypertext system [BIGE88] running under Microsoft Windows/386 on the IBM PS/2 Model 80. Later, the ITB was ported via a Guide hypertext system for the Macintosh IIcx. In the most current implementation, the ITB consists of five levels (see Figure 1). At the top is a display of the phase/method grid (Level I), with the phases of the system life cycle as they are shown across the top of the screen and the development approaches or methods shown down the left side.

To simplify the classification of these methodologies, they have been organized into three primary groups: Process Methodologies, Data Methodologies and Object-Oriented Methodologies. These groupings are described in greater detail in Appendix B.

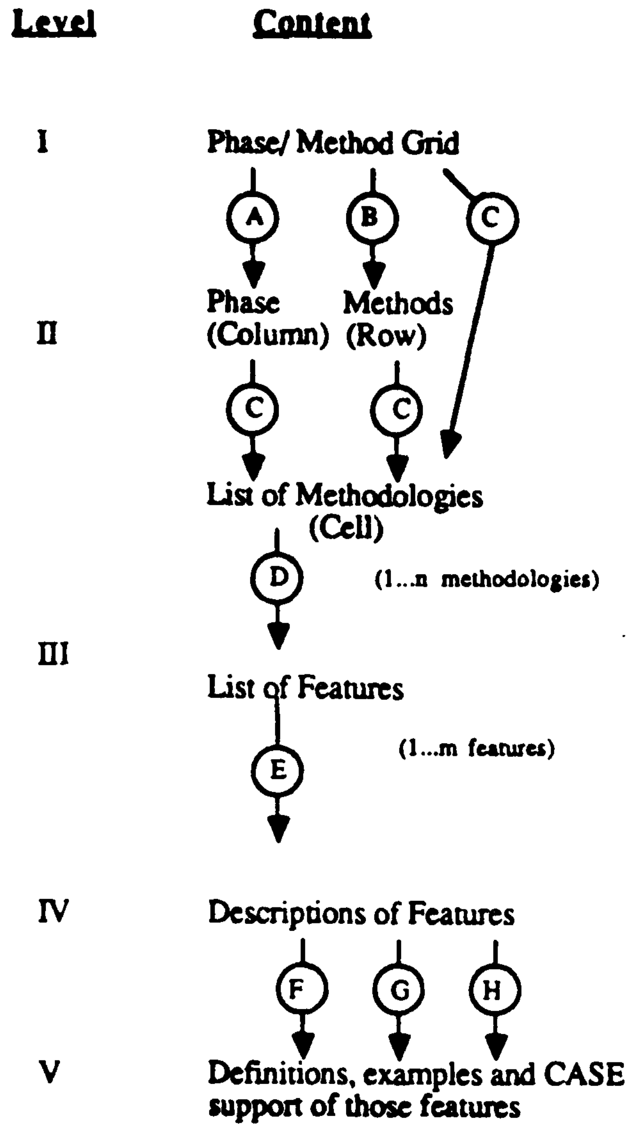

Figure 1. GSU ITB Schematic Overview

From the top level, additional information can be obtained on a particular phase (i.e., the column under "Feasibility") by selecting Route A and moving down to Level II. Similarly, information on a method (e.g., Process Methodologies) can be obtained through Route B; and a row/column cell is Route C. An example of such paths are shown in Figure 2.

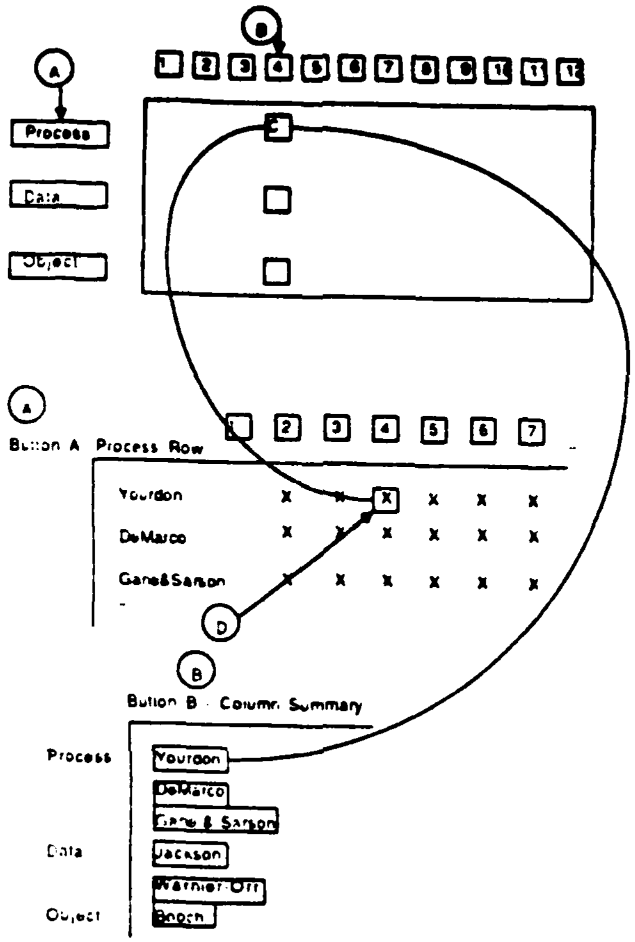

Figure 2. ITB, Levels I and II.

Once a particular methodology has been specified at Level II (e.g., Gane and Sarson), information on the features of this methodology can be obtained through Route D to Level III. Detailed descriptions of each of these features (e.g., Data Flow Diagrams) are found at the next level down, Level IV, through Route E as is shown in Figure 3.

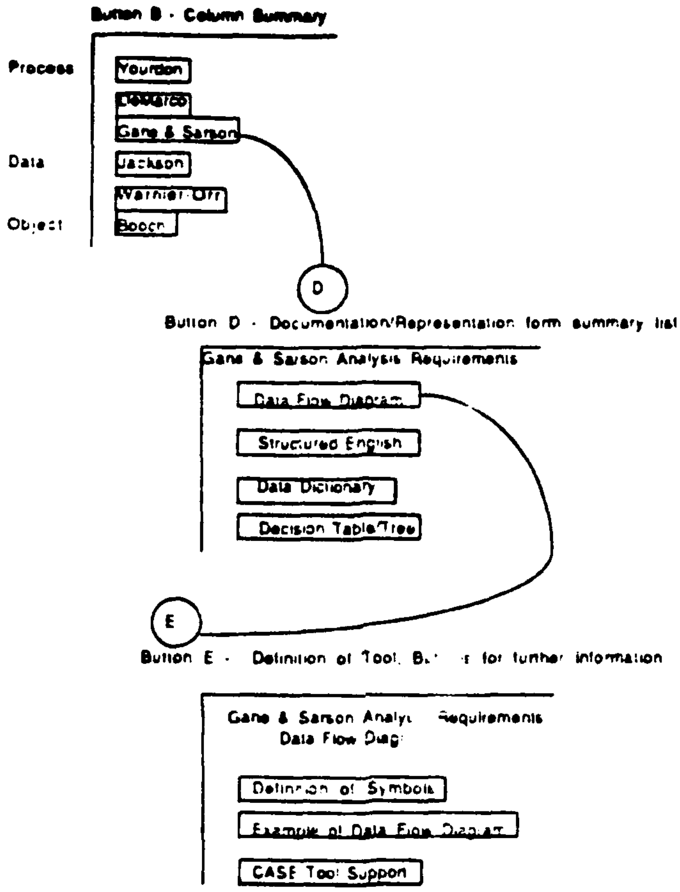

Figure 3. ITB, Levels III and IV.

Finally, at Level V which is the lowest level, information is available on specific feature definitions (Route F; see Figure 4), examples of use (Route G; see Figure 5), and listings of CASE tools that currently support this feature (Route H; see Figure 6). All of these steps are summarized in Table II.

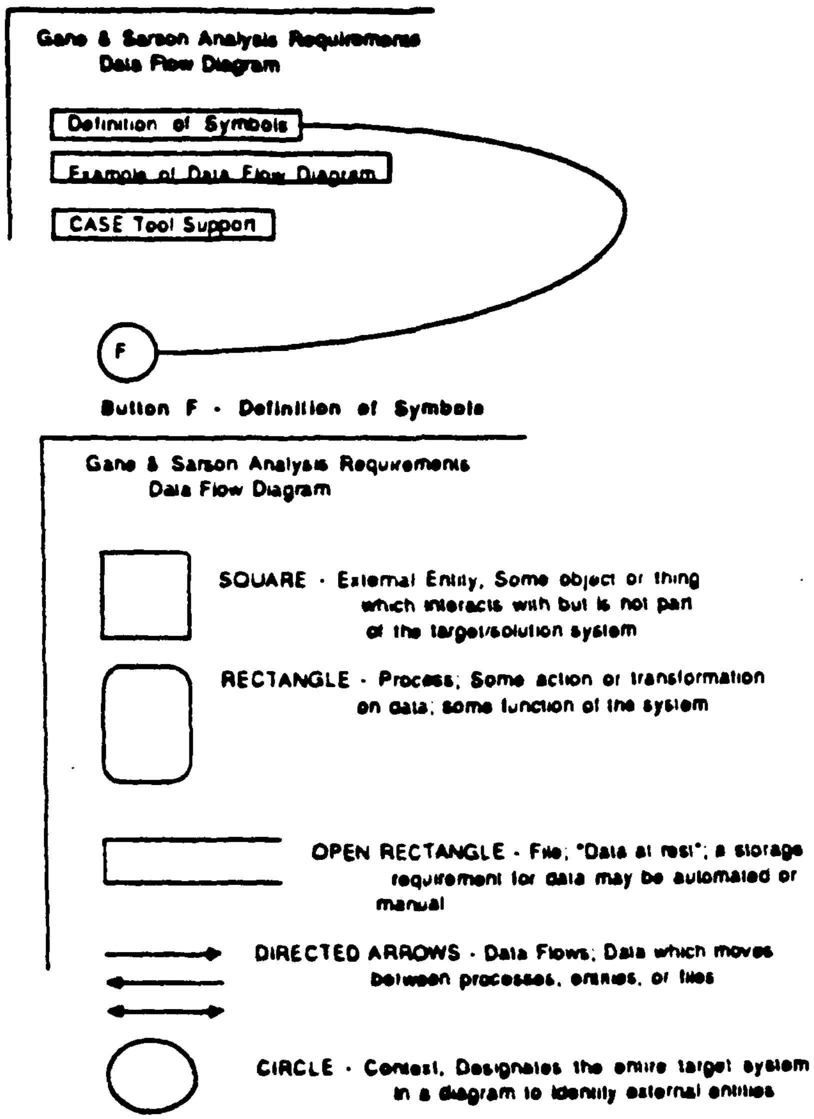

Figure 4. ITB, Level V showing F.

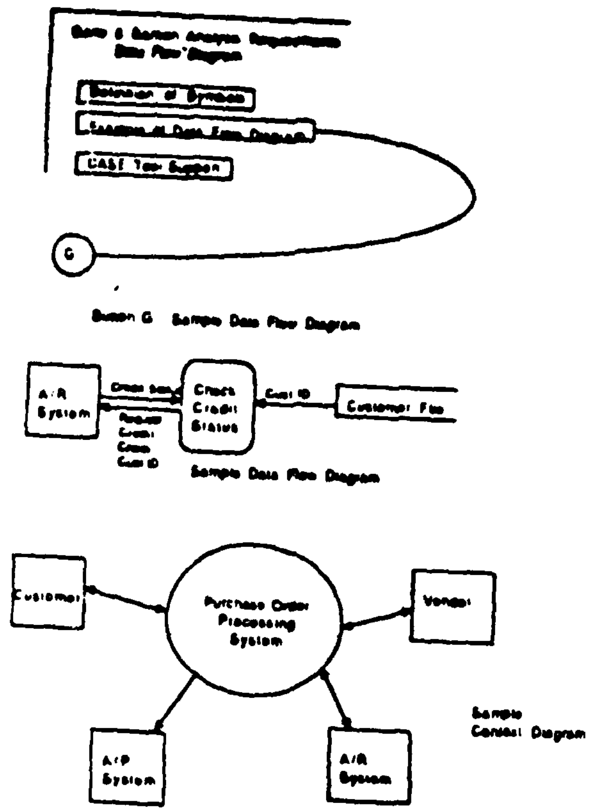

Figure 5. ITB, Level V showing G.

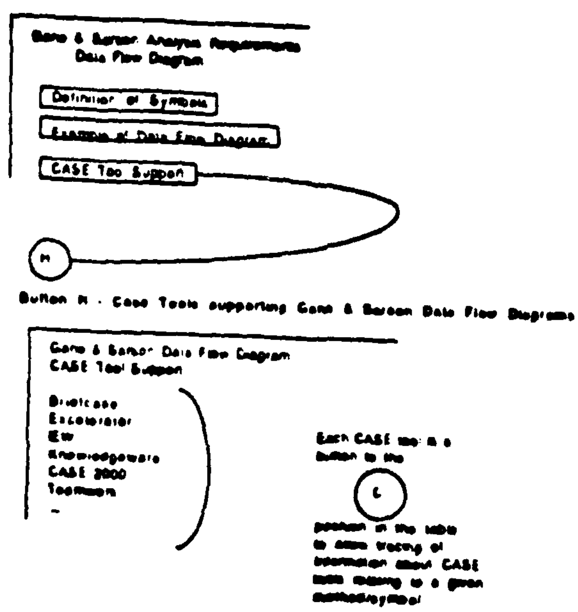

Figure 6. ITB, Level V showing H.

Table II. Overview of GSU ITB

Level	Description
I	Methodologies by Phase GSDLC
II	Initiated by clicking on a row. Major methods' authors (CASE Tools that support the Method)
II	Initiated by clicking on a column. Authors whose Method support that Phase (CASE Tools that support a Phase)
II	Initiated by clicking on row/column intersection. (Authors' Methods and CASE Tools for that Phase and that Method)
III	Initiated by clicking on a specific Method/CASE Tool. (For that Author/Tool, the Documentation Requirements and Graphic Representation Forms for a particular Author/Tool; e.g., Gane & Sarson/ Analysis: DFD, Data Dictionary, Structured English, Decision Table/Tree)
IV	Initiated by clicking on a specific feature. (Definition of Tool, buttons for further Information)
V	Initiated by clicking on "Definition of Symbols" (Symbol definition for representation form or document table of contents; e.g., Gane & Sarson: external entity - square, process - rounded rectangle, etc.)
V	Initiated by clicking on "Examples" (Example of representation form)
V	Initiated by clicking on "CASE Tool Support." (CASE tools - go to row/column intersection for CASE tools; e.g., Excelerator, IEW, KnowledgeWare, etc.)

Software Tools

The following section contains a discussion of the concepts underlying potential software development tools that could be introduced via the ITB. This is due to a lack of similar, available tools. Three specific prototype RSC tools are described: a Reuse Effort Assessment (REA) tool, a Closeness Measure Update (CMU) tool, and a Specifications Development (SD) tool [OWEN88b].

The REA tool is intended to advise on potential software reuse based on RSC size, structure, and documentation. The CMU tool would automate the process of deciding when closeness updates are needed. The SD tool would provide a means for transitions from informal to formal software requirements and specifications, ultimately producing a design, perhaps by the Object Oriented Development (OOD) method. The development of these tools would be accompanied by a comprehensive survey of existing non-Ada CASE tools as well as a compilation of prospective tools for the Ada MIS Environment.

Closeness Measure Update (CMU) Tool

The previously developed GSU RSC Management System (RSC_MGT System) is a working prototype implemented with a feedback feature that is used to modify the perceptual (conceptual) closeness measures for Reusable Software Components (RSC's). Although the RSC_MGT system collects the necessary data, the user still must decide when an update is needed and how to generate this update. A Closeness Measure Update (CMU) tool would automate this process as well as aid the user in deciding exactly when a closeness update is needed [GAGL88b]. One approach to the design of the CMU is the use of fuzzy logic to estimate the disparities between old and new closeness measures.

Reuse Effort Assessment (REA) Tool

The Reuse Effort Assessment (REA) tool would advise a potential software reuser on the tradeoffs between reusing a RSC versus developing a brand new software product. Essentially, the REA would provide a degree of difficulty of reuse. The final selections of RSC's could be provided through more easily interpretable degrees of memberships in the key RSC reusability attributes; e.g., size, structure, or documentation, etc., all of which would be weighted by reuser experience [OWEN88b]. The REA could also be based on fuzzy membership functions and fuzzy modifiers.

Specifications Development (SD) Tool

A Specifications Development (SD) tool is an extremely important tool in that it will: facilitate the creation, storage, and retrieval of RSC's; extend the RSC practical application from code to requirements, specifications, and design; and provide an interface to the RSC_MGT System. The SD tool can be developed independently of the GSU RSC_MGT System inasmuch as it is a general informal-to-formal specifications development tool. However, the RSC_MGT System, compared to, say, the RAPID Library System being developed by SofTech, is general purpose, runs on PC-class machines, is already operational, and is written in Ada.

The SD tool consists of four components: A Natural Language Interface; an Object Builder; a Specification Builder; plus a RSC_MGT System Interface. These software tools will contribute in the overall reusability effort by moving the emphasis "left" (or earlier) in the software life cycle. A major goal of this effort would be the demonstration of the reuse of requirements and specifications, and not just code. This tool could provide a better transition from informal to formal requirements and specifications. Thus, the SD tool would assist the user in developing formal requirements and specifications that lend themselves to automated methods, but that otherwise may be very difficult to generate or understand.

The user will be able to interact with the SD tool in limited natural language to produce, for example, an object-oriented design from which a set of formal specifications could be generated using VDM. Additional discussion on this and the two previous tools can be found in [GAGL88b, OWEN88b].

Future Activities

Extensions to the ITB

Several obvious extensions both to this research and the ITB are possible. If such extensions are made to the ITB, then it can be fully functional; i.e., intelligent. Thus, future versions of the ITB will be capable of addressing more complex tool-related questions. For instance, four representative types of questions would be as follows. First, if a need exists for a software tool, by what means can the best available tool be selected? Second, if a tool does not exist, how can its specifications best be obtained? Third, what could form the basis to guide the design of such a necessary but non-existent tool? Fourth, to what degree can the integration of existing tools be determined (e.g., whether or not they are "seamless")?

To answer such questions, several extensions to the ITB have been suggested. Three specific extensions are described below: an Intelligent Tutoring System (ITS) for OOD, a Functional Decomposition (FD) Tool, and a Software Productivity and Estimation (SPE) Tool.

Intelligent Tutoring System (ITS). One of the major problems that operating elements of the Information Systems and Engineering Command (ISEC) will face in the transition of many MIS applications from COBOL to Ada is the retraining of software personnel (e.g., designers, analysts, and programmers, etc.). Software development in Ada requires a somewhat different approach to software engineering than what has been done in the past. To facilitate such a transition, one possible extension of the ITB would be the ITS which could assist ISEC in teaching modern software engineering concepts, incorporating OOD and Ada.

A major difference between conventional CAI and the ITS is that the ITS would not only pose problems for the users to solve, but actively assist in the solution of the problem while also monitoring the student progress. The ITS can also offer advice either upon request or whenever an error is made.

As it is envisioned, the ITS would comprise of three parts: an expert system, a teaching model, and a user model. The expert system could solve problems posed by the user. Interfaces and dialogue between the system and the user would be accomplished in limited domain natural language. The teaching model would combine the pedagogical method chosen, as well as the substantive material, for a given subject area. The user model would monitor the user, assess the user's current progress, and provide advice as necessary.

An OOD tutorial, for example, would present a typical application problem in natural language description. In accordance with OOD procedures, it would then ask the user to determine: the objects, their attributes, the operations suffered by the objects, the attributes of the operations and the dependencies of the objects. The output of this process would contain both a textual as well as graphical OOD description of the system. If the problem is sufficiently complex, then the process would recursively determine the required sub-objects until a complete OOD design is produced.

For each stored problem, the OOD tutorial would use the respective solution to monitor the user's progress as being developed in OOD. The user can ask for advice at any point or, if the user makes an error, the system would interrupt with advice which would be context sensitive.

Based on the OOD tutorial described above, the fully developed ITB would be a very practical tool with the capability for the user to enter a problem description, rather than using the stored problems, and the system itself would generate a solution. At this point, the system could become a true design tool rather than simply an intelligent tutor.

Functional Decomposition (FD) Tool. Functionality is a means of creating RSC's and achieving reusability. Tools to decompose processes and describe their functionality are needed before Ada RSCs can be developed, particularly for environments such as STAMIS. The central idea here is that common functions from different STAMIS systems, if carefully defined and sufficiently repeated, could form a basis for reusability.

The method behind the FD tool was presented at the February 1988 ACM Atlanta Computer Science Conference, and a description is contained in the Conference Proceedings [GAGL88a]. The basis of the FD tool is a resource management model which provides the means to capture the essence of functionality. The structure for the resource management model assumes the form of a directed graph (digraph) of entities as nodes with identical activities as arcs at each of the nodes. From this (first) digraph of entities, a second digraph of common functions is derived whose nodes are then decomposed to identify unique software modules. Subsequent applications of structured design lead to the internal organization of each of the modules.

Like other large MIS, the STAMIS are mainly concerned with personnel, resource acquisition, management, distribution, transportation, and associated financial activities. Their various software systems exhibit files of such variety that common features are often obscured. Nevertheless, the commonalities that exist could be uncovered and subsequently utilized by this functional decomposition tool.

The transition from the entity digraph to the functional digraph would then be instantiated with this tool. Whereas the overall intent of this tool is to assist in the task of producing RSC's, the specifics of any application would depend on the identification of the nodes and connections, as well as the structure of the entity digraph.

Moreover, the transition to the functional digraph would be made by using a top-down approach beginning with the most aggregated view of the activities involved. Then, successively more disaggregated digraphs can be constructed using OOD. The process is stopped when the benefits of further disaggregation cannot be justified through reusability.

This FD tool would employ an inductive scheme to find the appropriate abstraction concepts, and a deductive scheme to show how the software specifications can be developed based on functionality. The frequency of calls as indicated by the arcs in the functional digraph would provide an a priori measure of reusability of the various modules.

Software Productivity and Estimation (SPE) Tool. A key dimension in software development is productivity, with major questions concerning how to define, measure and improve it. One definition, in the usual sense of efficiency, holds that productivity is the ratio of output to input. Here, "input" means the resources consumed in the creation of the system (time of the individuals involved and the cost of the computer resources employed). Such times and costs can be measured in a fairly straightforward fashion after the fact. However, they are much more difficult to estimate in advance when perhaps only the specifications are available.

Of even greater difficulty is the measurement of "output," either before or after the system is operational. Such well established metrics as lines of code (LOC) are notoriously unreliable. For instance, if the same system were to be developed in a third generation language (e.g., COBOL or Ada) versus a fourth generation language (e.g., Focus or RAMIS), LOC comparisons would be meaningless, even though both versions contained the same functionality.

In commercial environments, the use of Function Point Analysis (FPA) as both an estimating tool and as a gauge of productivity is beginning to gain many converts. However, virtually nothing has been done with this approach in an Ada environment. The development of a prototype of this tool within the ITB would pinpoint the similarities and differences between such environments. More importantly, the calculation of a variety of different function points could be adapted to the Army MIS redirection efforts.

The use of FPA would require careful calibration at each development site in order to be truly useful. Analysis of the inputs, outputs, files, interfaces, and inquiries involved in a system, coupled with measures of complexity specific to a project and its development environment, could result in metrics that are of high value in a number of different aspects. First, such metrics could be used before system construction to estimate project costs. Second, they could be used to compare different development approaches and CASE tools. Third, they can be used to compare and track the productivity of different development sites over time.

Each of these kinds of uses of FPA should be of great potential benefit to the Army and other parts of DoD, particularly during times of ever tightening budgets, in the effort to improve cost effectiveness of software development and maintenance.

Software Tools Survey

Several times in this report, a need for a software tools survey has been indicated. Our approach to such a survey would be a classification defined by four classes of tools. These classes are: (1) commercial CASE tools in-use by the Standard Army Management Information Systems (STAMIS), (2) those tools not currently in use but which are STAMIS-particular tools, (3) commercially available CASE tools that are used by STAMIS, and (4) other prospective tools.

An important continuation of this project would utilize the classification scheme above to conduct a software tools survey in two phases. The first phase would identify the first two classes of tools and how such tools are used. It is presumed that much of this type of data pertinent to the STAMIS could be made available to AIRMICS through internal self-assessment studies. Knowledge about tools obtained from this phase could also then be entered into the ITB for analysis.

The second phase of the survey extension would determine the third class of tools, and that information similarly could be entered into the ITB. The results could then be compared, using the functions supplied by the ITB, to identify and describe the fourth class of tools that could be integrated into STAMIS software development and maintenance.

<u>Summary</u>

In this thirteen-month, coordinated yet multi-faceted research effort, GSU has investigated issues in the analysis, selection, implementation, and evaluation of software tools. Several significant contributions have been made, the most important of which is the development of a prototype Intelligent TestBed or ITB. The prototype was implemented on both an IBM PS/2 and a Mac IIcx, with detailed discussions of its use provided in appendices C and D. The major deliverables of this project are, therefore, these two versions of the ITB, the associated user documentation, copies of a videotape of a demonstration session, and two conference papers.

This final technical report contains a brief review of some of the issues involved in software tools, reuse, and libraries, plus an overview of this project as well as the origins of the ITB. However, the major discussion concerns the development and demonstration of the ITB which currently is only minimally populated with descriptions of software development methodologies and references to CASE tools. An immediate extension to the ITB would allow the continuing classification of software tools, as well as the testing of other tool "proofs of concept." Additional potential uses of the ITB have also been suggested, for example, the ITB tutorial system, the use of the ITB for PC-based Ada tools tutorials, and the software tools survey.

The ITB was designed to provide a computer generated medium in which to analyze software development methodologies and allow the visualization of their organizing schema. Its two dimensions are as follows: the columns turned out to be the phases of Georgia State Development Life Cycle (GSDLC) and the rows are different software development methodologies. Entries in the five levels of the ITB can be accessed by use of hypertext buttons.

The ITB is conceptually different from, and perhaps superior to, either a PDL or an SDE such as Methodology Engineering. The ITB does not attempt to formally describe methodologies, nor does it provide the means for generating or adapting customized techniques for SDEs. Additionally, the ITB is adaptable to a three levels of users: the <u>novice</u> for inquiring of information; <u>management</u> for gaining directions of intent; and, experienced software <u>designers and programmers</u> for assessing tradeoffs of methods and tools.

References

[BJOR78] Bjorner, D., "Software Abstraction Principles," in Lecture Notes in Computer Science 61: The Vienna Development Method: The Meta Language. Bjorner, D., and C. B. Jones (eds.), Springer-Verlag, New York, 337-374, 1978.

[BOEH81] Boehm, B. Software Engineering Economics. Prentice-Hall, 1981.

[CONG89] Conger, S. A. and J. L. Wynekoop, "A Framework for Evaluating Computer Aided Software Engineering Research Tools," Proceedings of the ACM Southeast Regional Conference, April 1989.

[DAVI85] Davis, G. B. and M. H. Olson. Management Information Systems: Conceptual Foundations, Structure and Development. McGraw-Hill, 1985.

[D2167A] DoD Mil-Std 2167A, Software Development (draft), 1988.

[BIGE88] Bigelow, J., "Hypertext and CASE", IEEE Software, Vol. 5 (2), 23-27, March, 1988.

[DUCE88] Duce, D. A., Fielding E. V. C., and Marshall, L.S., "Formal Specification of a Small Example Based on GKS", ACM Transactions on Graphics, Vol. 7, No. 3, 180-197, July 1988.

[FREE80] Freeman, P. and Wasserman, A. A Tutorial on Software Design Techniques, 3rd Edition, IEEE Computer Society, 1980.

[GAGL88a] Gagliano, R. A., M. D. Fraser, M. E. Schaefer and G. S. Owen. "Functionality in the Reusability of Software", Proceedings of the ACM 88 Computer Science Conference, 540-545, February 1988.

[GAGL88b] Gagliano, R. A., M. D. Fraser, G. S. Owen, and P. A. Honkanen, "Tools for the Classification, Storage, and Retrieval of Reusable Ada Software," (forthcoming) Proceedings of the 6th Symposium on Empirical Foundations of Information and Software Sciences, Atlanta, GA, October 1988.

[GAGL89] Gagliano, R. A., M. D. Fraser, and G. S. Owen. "Guidelines for Reusable Ada Library Tools," in Ada Reusability Guidebook, Project DE-AC05-840R21400, Martin Marietta Energy Systems, Inc., 83-94, 1989.

[IEEE83] IEEE Software Engineering Dictionary. IEEE, 1983.

[JONE78] Jones, C. B. (1978) "The Meta-Language: A Reference Manual," in Lecture Notes in Computer Science 61: The Vienna Development Method: The Meta Language. Bjorner, D. and C. B. Jones (eds.), Springer-Verlag, New York, 218-277.

[KUMA88] Kumar, K. and R. J. Welke, "Methodology Engineering: A Proposal for Situation Specific Methodology Construction," in the Proceedings of the Case Studies 88 Conference, Ann Arbor, 1988.

[OWEN87] Owen, G. S., R. A. Gagliano, and P. A. Honkanen, "Functional Specifications of Reusable MIS Software in Ada", Proceedings of the Joint Ada Conference 5th National Conference on Ada Technology and Symposium, 19-26, March 1987.

[OWEN88a] Owen, G. S., R. A. Gagliano, and P. A. Honkanen, "Tools for the Storage and Retrieval of Reusable MIS Software in Ada," Proceedings of the ACM 88 Computer Science Conference, 535-539, February 1988.

[OWEN88b] Owen, G. S. , M. D. Fraser and R. A. Gagliano, "Knowledge Based Tools for Reusable Ada Software," (forthcoming) Proceedings of the 6th Symposium on Empirical Foundations of Information and Software Sciences, Atlanta, GA, October 1988.

[SCHA88] Schafer, G. Functional Analysis of Office Requirements: A Multiperspective Approach. New York: John Wiley, 1988.

[SUYD87] Suydam, W., "CASE Makes Strides Toward Automated Software Development", Computer Design, Vol. 26, No. 1, 49-70, January 1, 1987.

[WELK83] Welke, R. J., "IS/DSS: DBMS Support for Information Systems Development," in C. W. Holsapple and A. B. Winston (eds.), Data Base Management: Theory and Applications. D. Reidel Publ., 195-250, 1983.

Appendices

A. Ada Technology Conference Paper #1
("A Structured Stepwise Refinement Method for VDM")

B. Ada Technology Conference Paper #2
("The Intelligent Test Bed: A Tool for Software Development and Software Engineering Education")

C. ITB User Manual for the PS/2

D. ITB User Manual for the Mac IIcx

E. ITB Specification in VDM